Санкт-Петербургский государственный университет
Кафедра компьютерного моделирования и многопроцессорных систем

Джанелидзе Григорий Михайлович

Выпускная квалификационная работа бакалавра

Рекуррентные нейронные сети в задаче анализа тональности текста

Направление 010300
Фундаментальная информатика и информационные технологии

Научный руководитель,
доктор физ.-мат. наук,
профессор
Андрианов С.Н.

Санкт-Петербург
2016

Содержание

Введение

За последние десятилетие количество пользоваетелей сети Интернет увеличилось в 3 раза[1]. За это время Интернет сильно преобразился — на место статичным страницам, редактируемым одним человеком, пришли страницы, где любой человек может что-то написать, что-то прокомментировать — грань между потребителями и создателями контента начала стираться.

Пользователи создают огромное число различных материалов: записи в форумах и блога, «статусы» в социальных сетях, фотографии и видео на специализированных ресурсах и многое другое. С течением времени появилось несколько крупных сайтов, собирающих мнения про, например, товары, фильмы, книги; вот некоторые из них: Amazon[2], Metacritic[3], IMDB[4]. Некоторые из них даже входят в 500 самых посещаемых сайтов[5] сети Интернет. Прежде чем отдать предпочтение одному из товаров, у человека появилось возможность узнать мнение остальных людей, которые владеют этим товаром. Количество таких мнений огромно, что позволяет человеку узнать все необходимое и лишь затем сделать выбор. Но с течением временеми отзывов стало слишком много — изучить все отзывы на какой-либо популярный товар является невыполнимой задачей для одного человека. Для производителей товаров это стало особо важно — иногда есть необходимость следить за отзывами, а держать большой отдел экспертов-лингвистов, который осуществлял бы мониторинг отзывов и оценивал их — экономически невыгодно. Эта ситуация является причиной появления такого предмета, как анализ мнений: была острая необходимость в создании системы классификации мнений с минимальным участием людей.

Анализ мнений включает в себя несколько задач, направленных на решение одной и той же проблемы — качественная оценка отношения автора текста к субъекту, который рассматривается в этом же тексте. Из конкретных задач, которые входят в анализ мнений, можноо выделить такие задачи, как: оценка субъективности или объективности текста по отношению к субъекту, классификация отношения автора к субъекту. Последняя задача называется задачей анализа тональности текста и является фундаментальной в анализе мнений — чтобы делать какие-то дальнейшие выводы про мнение, необходимо сперва понять, как автор относится к субъекту.

Задача анализа тональности текста сводится к задаче класификации. Текст нужно отности к одному из классов эмоциональной окраски, например, «положительный», «отрицательный» или «нейтральный». В общем

[1] По исследованию Statista: `http://www.statista.com/statistics/273018/number-of-internet-users-worldwide/`
[2] `https://amazon.com`
[3] `http://www.metacritic.com/`
[4] `http://imdb.com`
[5] По данным Alexa: `http://www.alexa.com/topsites`

случае, число классов конечное. Классификаторы могут работать как с исходными данными(например, целым текстом, отдельными предложениями, отдельными словами текста, n-граммами), так и с некоторыми векторными представлениями текста или слов(например, мешок слов [1]). Иногда классификация происходит в два этапа и на обоих этапах является бинарной. На первом отделяются субъективные сообщения от объективных. Объективными в этом случае называются как раз те, которые не несут эмоциональной окраски и являются нейтральными в варианте с тремя классами. Второй этап делит субъективные тексты на положительные и отрицательные. Получается, что решение задачи тональности текста с 3 классами, решает в один этап и задачу оценки субъективности или объективности.

За последние годы искуственные нейронные сети доказали свою эффективность для многих задач. Так, нейронные сети могут выделять отдельные гитарные аккорды в музыке [2], распознавать цифры [3] и другие объекты на изображениях [4].

Искусственной нейронной сетью называется математическая модель, построенная по принципу организации и функционирования биологических — сетей нарвных клеток живых организмов. Это модель была получена в середине XX века при попытке смоделировать процессы, протекающие в человеческом мозге. Искусственная нейронная сеть состоит из соединенных между собой «нейронов», каждому из которых присвоена определенную функцию активации. По соединениями между нейронами распространяется сигнал. Каждое соединение между нейронами имеет численное значение, которое называется весом. Состояние нейрона определяется по формуле $S = \sum_{i=1}^{n} x_i w_i$, где n — число входов нейрона, x_i — значение входа i-ого нейрона, w_i — вес у i-ого входа. Для дальнейшего распространения сигнала, над состоянием применяется функция активации и полученное значение передается остальным нейронам. Визуально нейронную сеть можно представить в виде взвешенного направленного графа, где вершины будут нейронами, а ребра — связями. Процесс обучения нейронной сети заключается в нахождении всех весов связей.

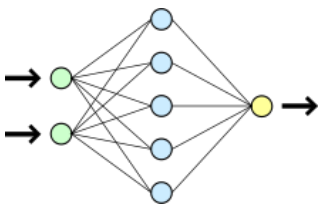

Рис. 1: Двухслойный перцептрон с 2 входными и 1 выходным нейронном. Зеленым цветом обозначены входные нейроны, голубым — скрытые нейроны, желтым — выходные нейроны.

Нейронные сети классифицируют на 2 класса: нейронные сети прямого распространения сигнала и нейронные сети с обратными связями. Одним из наиболее распространенных типов нейронной сети прямого распростра-

нения сигнала является многослойный перцептрон, его пример представлен на рисунке 1. К нейронным сетям с обратными связями относят рекуррентные нейронные сети. Рекуррентные нейронные сети показывают хорошие результаты во многих задачах связанных с классификацией последовательностей, например, распознание речи [5] и подавление шума [6]. Помимо этого, рекуррентные нейронные сети показывают одни из лучших результатов на данный момент в некоторых задачах связанных с обработкой текста на естественном языке, например, в задаче классификации текста [7].

Целью данной работы было применение рекуррентных нейронных сетей к задаче анализа тональности текста и сравнение полученных результатов с результатами существующих подходов к этой задаче.

Постановка задачи

Целью данной работы является создание модели рекуррентной нейронной сети для решения задачи анализа тональности текста. В данной работе рассмотрена бинарная классификация на 2 класса — «положительный отзыв» и «отрицательный отзыв».

Для достижения заданной цели необходимо:

1. Проанализировать существующие методы решения задачи анализа тональности текста.
2. Обосновать, описать и реализовать архитектуру рекуррентной нейронной сети.
3. Сравнить полученную модель с существующими методами решения.

Обзор литературы

Впервые задача анализа тональности текста была решена вычислительно методами машинного обучения в 2002 году. В публикациях были рассмотрены решения методом обучения без учителя [8] и методом обучения с учителем [9]. В этих статьях рассматривались отзывы на различные товары: была поставлена задача выяснить, рекомендует или нет автора отзыва товар, которому посвящен отзыв. В этих работах была сформулирована задача, сформирована основная терминология и предложены первые варианты решения данной задачи. В работе [10] была предложена модификация наивного байесовского классификатора, которая позволила улучшить результаты, полученные в [9].

По использованию нейронных сетей в различных задачах обработки текстов на естественном языке существует много работ. В исследовании [7] было предложено использовать сверточную нейронную сеть для классификации текста. Получившиеся в этой работе результаты стали лучшими достигнутыми на данный момент. В работе [11] на основе свёрточных нейронных сетей предложена архитектура нейронной сети для анализа тональности коротких текстов(до 140 символов). В исследовании [12] анализируются приемущества использование функции активации $f(x) = max(0, x)$, для сравнения разных функций активации берется задача анализа тональности текста. Полученные результаты показывают небольшой выигрыш при использовании этой функции активации.

Глава 1. Существующие подходы к задаче анализа тональности текста

1.1. Методы обучения без учителя

Методы обучения без учителя заключаются в том, что алгоритмы работают с неразмеченными данными. Их принадлежность к какому-либо из классов алгоритмы должны определить сами, например, по структуре этих данных. Основное отличие от остальных методов заключается в том, что при обучении без учителя правильность классификации не влияет на процесс обучения. Самый распространенный пример такого метода — кластерный анализ.

В статье [8] был предложен алгоритм обучения без учителя для классификации мнений на 2 категории «рекоммендует» и «не рекоммендует». В алгоритме есть три этапа:

1. Поиск би-граммов, в которые входят наречия и прилагательные. Одно из слов служит контекстом, второе — его описанием.
2. Определение тональности словосочетания: «хорошо» или «плохо».
3. Определение тональности отзыва. На данном этапе подсчитывается средняя тональность всех словосочетаний, найденных на первом шаге. Если среднее получилось положительным, то отзыв относится к «рекоммендует», иначе — к «не рекоммендует».

Алгоритм достигает точности в 80% на отзывах, в которых содержится несколько предложений. У алгоритма так же есть один большой недостаток — для второго шага необходим корпус, который собирают эксперты-лингвисты вручную. Получается, что алгоритм привязан к конкретному языку и в нем участвует человеческий фактор. Это создает отдельную проблему при анализе мнений из сети Интернет — в корпусе нужно учитывать возможные ошибки и опечатки пользователей.

1.2. Методы обучения с учителем

Методы обучения с учителем позволяют классфицировать данные на основании какого-то заранее классифицированного набора данных, называемого тренировочным набором. Такие методы должны реализовывать две функции: обучение на тренировочном наборе данных и осуществлять классификацию новых данных. Обучением называется процесс построения или же аппроксимации функции, которая позволит обобщить классификацию на все данные так, чтобы эта классификация была наиболее близка к действительной.

Методы обучения с учителем подвержены проблеме переобучения. Переобучение — это явление, когда построенная модель хорошо классифицирует примеры из тренировочного набора, но относительно плохо классифицирует любые другие примеры. Это явление происходит, потому что в процессе обучения модель выявляет некоторые закономерности в тренировочном наборе, которые отсутствуют в генеральной совокупности. Способы борьбы с переобучением зависят от конкретных методов и моделей.

1.2.1. Наивный байесовский классификатор

Наивный байесовский классификатор [13], как очевидно из названия, использует теорему Байеса — оперирует условными вероятностями. «Наивный» означает, что делается предположение о том, что слова в предложении не являются зависимыми. Этот классификатор крайне прост, но даже несмотря на это показывает себя очень эффективным в решении задач классифкации текстов [14].

Для построение наивного байесовского классификатора необходимо выбрать закон, по которому распределены данные. Обучением данного классификатора является вычисление параметров распределения по примерам из тестового набора данных.

если предположить, что данные распределены по закону Бернулли[6], то класс c^*, к которому относится текст t, вычисляется следующей формулой:

$$c^* = \operatorname*{argmax}_c P(c) \sum_{i=1}^{m} P(x|c)^{x_i(t)}$$

Здесь x — характеристики, по которым оцениваются тексты, их количество равно m, $x_i(t)$ — величины, показывающие наличие i-ая характеристики в тексте, а c — метка класса. $P(c)$ и $P(x|c)$ параметры, вычисленные при обучении классификатора.

1.2.2. Метод максимальной энтропии

Метод максимальной энтропии [15] в случае бинарной классификации представляет собой логистическую регрессию. В отличии от наивного байесовского классификатора метод максимальной энтропии учитывает зависимость признаков. Поэтому для классификации можно использовать, например, n-граммы и словосочетания одновременно.

Модель классификатора описывается формулой

$$P(c|t,\lambda) = \frac{\exp(\sum_{i=1}^{N} \lambda_i x_i(c,t))}{\sum_c \exp(\sum_{i=1}^{N} \lambda_i x_i(c,t))}$$

[6]Распределение Бернулли имеет плотность $p(k|\theta) = \theta^k(1-\theta)^k$, где $k \in \{0,1\}$, $\theta \in [0,1]$.

Здесь c —- метка класса, t — классифицируемый текст, $x_i(c,t)$ — совместная представленность i-ого признака в классе c и в объекте t, N — количество признаков, λ — вектор весов для всех признаков, который отвечает за релевантность признаков. Процесс обучения заключается в нахождение вектора λ.

Класс c^*, к которому относится текст t для модели с параметрами λ нужно, вычисляется следующим образом:

$$c^* = \operatorname*{argmax}_c P(c|t,\lambda)$$

1.2.3. Метод опорных векторов

Метод опорных векторов [16] основан на разбиении пространства классифицируемых данных на подпространства, где каждому подпространству соответствует один и только один класс. Данному методу необходим этап предобработки данных, чтобы преобразовать текстовые данные в числовые векторы. В процессе обучения над пространствами осуществляются преобразования с помощью оператора ядра такие, чтобы данные можно было разделить гиперплоскостями на подпространства, в которых все или почти все данные из тестового набора принадлежат подпространству своего класса. Операторы ядра могут быть как линейные, так и нелинейные. Затем при классификации нового текста он также переводится в векторное представление, после чего ему присваивается класс, в зависимости от подпространства, где находится полученный вектор.

Процесс разбиения при бинарной классификации при помощи линейного оператора ядра проиллюстрирован на Рис. 2.

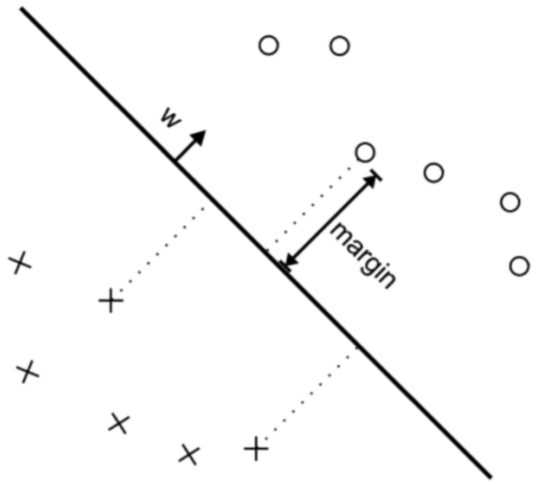

Рис. 2: Классификация на 2 класса методом опрных векторов с линейным оператором ядра. $margin$ —- расстояние до гиперплоскости, w — классифицируемый вектор.

1.3. Анализ и сравнение методов

Для сравнения методов был выбран набор данных Large Movie Review Dataset [17]. Этот набор состоит из 25 тысяч примеров для обучения и 25 тысяч примеров для тестирования, все примеры размечены на 2 класса — положительные и негативные. Набор собран в 2011 году из отзывов на фильмы, причем в наборе количество отзывов на один и тот же фильм не превышает 30 — чтобы при обучении модели на результатты её классификации не виляли имена актеров, режиссеров и другие факторы, связанные с конкретными фильмами. Набор содержит отзывы только на английском языке. Для сравнения алгоритмов использовалась их реализация на языке Python из библиотеки scikit-learn [18]. Для веторных представлений слов используется модель мешок слов. В таблице 1 представлены результаты работы классификаторов, рассмотренных в этой главе.

Для оценки качества классификации используется $F_1 - \text{score}$ — гармоническое среднее двух величин: precision и recall. $F_1 - \text{score}$ вычисляется следующей формулой:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

где precision — вероятность того, что случайно выбранный отзыв из какого-либо класса модель отнесет к тому классу, к которому отзыв действительно принадлежит; recall — вероятность того, что выбранный отзыв из класса при класификации будет отнесен к тому же классу. Более формально эти величины можно определить следующим образом: пусть при бинарной классификации есть множество всех классифицируемых текстов $\mathcal{F}$ и есть два класса c_1 и c_2. Обозначим за $\mathcal{F}_{c_1}$ множество текстов, которые относятся к классу c_1, а за $\mathcal{F}_{c_2}$ множество текстов, которые относятся к классу c_2, причем $\mathcal{F} = \mathcal{F}_{c_1} \cup \mathcal{F}_{c_2}$ и $\mathcal{F}_{c_1} \cap \mathcal{F}_{c_2} = \varnothing$. Тогда можно ввести следующие величины:

- T_p – true positives(истинно-положительные) – количество текстов из $\mathcal{F}$, которым модель присвоила класс c_1.
- F_p – false positives(ложно-положительные) – количество текстов из $\mathcal{F}_{c_2}$, которым модель присвоила класс c_1.
- F_n – false negatives(ложно-отрицательные) – количество текстов из $\mathcal{F}_{c_2}$, которым модель присвоила класс c_2.
- T_n – true negatives(истинно-отрицательные) – количество текстов из $\mathcal{F}_{c_1}$, которым модель присвоила класс c_2.

В этих величина precision и recall определяются следующими форму-

Классификатор	F_1**-score**	**Время обучения**
Найвный байесовский классификатор	0.78	21 секунда
Логистическая регрессия	0.80	1310 секунд
Метод опорных векторов	0.80	2231 секунда

Таблица 1: Сравнение подходов к задаче анализа тональности текста

лами:

$$\text{precision} = \frac{T_p}{T_p + F_p}$$

$$\text{recall} = \frac{T_p}{T_p + F_n}$$

Как видно из таблицы 1 методы показали примерно одинаковое качество классификации, отличия есть лишь во времени обучения[7]. Для анализа тональности важную роль играет именно $F_1 - \text{score}$, поэтому для дальнейшей работы стоит ориентироваться именно на улучшение данной величины, а не на уменьшение время обучения.

[7]Измерения проводились на процессоре Intel Core i5-4278U. Время затраченное на ввод-ввывод, а во время работы программы все данные хранились в оперативной памяти.

Глава 2. Архитектура нейронной сети

2.1. Рекуррентные нейронные сети

Рекуррентные нейронные сети — подкласс неройнных сетей с обратными связями, которые используют предыдущие состояния сети для вычисления текущего [19]. Простейший пример такой сети представлен на рисунке 3. Проводя аналогии с человеческим мозгом, можно сказать, что рекуррентные нейронные сети добавляют «память» к искуственным нейронным сетям. Обычно такой класс нейронных сетей используется в задачах обработки последовательностей нефиксированной длины. В частности рекуррентные нейронные сети показывают результаты лучше других методов в задачах классификации текста [7].

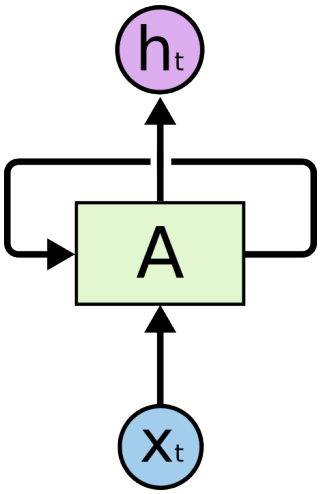

Рис. 3: Архитектура рекуррентной нейронной сети с одним входным нейронном, одним скрытым нейроном и одним выходным нейроном

Процесс обучения рекуррентной нейронной сети происходит следующим образом: на каждом шаге обучения t значения скрытого слоя нейронной сети $k_t \in \mathbb{R}^m$ вычисляется по формуле

$$k_t = f(Wx_t + Uk_{t-1} + b_k)$$

В этой формуле $x_t \in \mathbb{R}^m$ — входной вектор в момент времни t; $W \in \mathbb{R}^{m \times n}$, $U \in \mathbb{R}^{m \times m}$ и $b_k \in \mathbb{R}^m$ — параметры нейронной сети, которые вычисляются при обучении; f — функция активации.

В простой рекуррентной нейронной сети, которая представлена на рисунке 4, выходное значение $h_t \in \mathbb{R}^l$ на шаге t вычисляется по формуле

$$h_t = W_c k_t + b_c$$

.

Но память, реализуемая такой архитектурой, получается довольно

короткой — при каждой итерации информация в памяти смешивается с новой информацией и через несколько итераций полностью перезаписывается. На примере задачи анализа тональности текста это особенно важно, поскольку слова, стоящие в начале в длинном тексте, не будут вносить какой-либо учитываться в результатах работы нейронной сети ближе к концу текста, а ошибка на первых словах текста или предложения будут сильно влиять на общую ошибку нейронной сети. Эта проблема называется проблемой исчезающего градиента(vanishing gradient problem), в конце XX-го века она стала причиной застоя в исследованиях нейронных сетей — иследователи переключились на метод опроных векторов. Эта проблема характерна как для нейронных сетей прямого распространения, так и для нейронных сетей с обратными связями.

Для проблемы исчезающего градиента нет универсального решения — конкретное решение будет зависеть от архитектуры нейронной сети, поэтому эта проблема и по сегодняшний день остается актуальной. Подробнее про проблему исчезающего градиента можно прочитать в работе [20]. Проблему исчезающего градиента в рекуррентных нейронных сетях посвящена работа [21].

Для обучения рекуррентных нейронных сетей используется алгоритм обратного распространения ошибки по времени(backpropagation through time) [22], который является вариантом алгоритма обратного распространения ошибки(backpropagation), используемого для нейронных сетей прямого распространения сигнала. На рисунке 5 псевдокодом изображен данный алгоритм. Остановочное условие обычно выбирается из двух: либо нейронная сеть обучается заданное количество итераций; либо используется метод ранней остановки. Метод ранней остановки заключается в том, что есть верхнее ограничение по количеству итераций обучения и вводится дополнительный проверочный набор данных, который после каждой итерации обучения классифицируется нейронной сетью. Если результаты становятся хуже или не меняются на протяжении нескольких итераций, то обучение считается завершенным.

Поскольку обратное распространение ошибки по времени использует внутри себя обычный метод обратного распространения ошибки, то в нем применимы те же методы для избежания проблемы переобучения.

2.2. Архитектура «долгая краткосрочная память»

Одним из решений проблемы исчезающего градиента в рекуррентных нейронных сетях является архитектура «долгая краткосрочная память»(LSTM — long short-term memory) [23]. Долгая краткосрочная память является разновидностью рекуррентных нейронных сетей, схема такой сети представлена на рисунке 6.

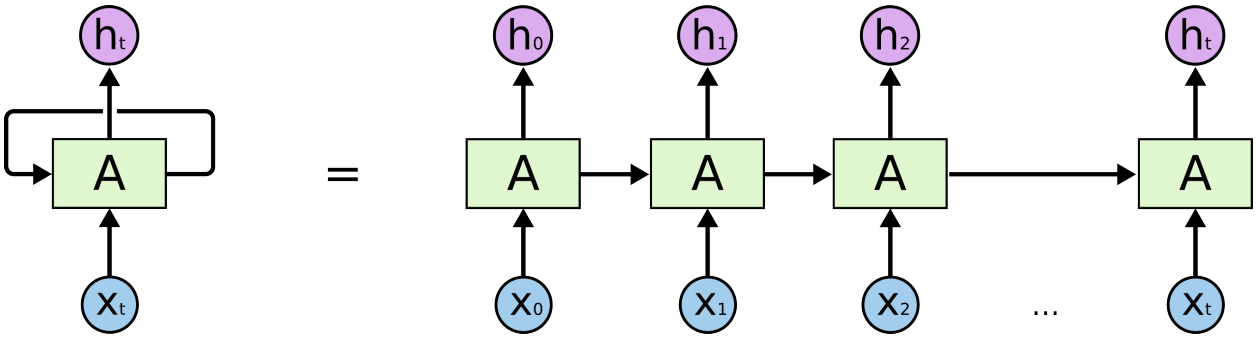

Рис. 4: Архитектура рекуррентной нейронной сети с одним входным нейронном, одним скрытым нейроном и одним выходным нейроном и ее развернутое по времени представление

Data: Массив a, где $a[t]$ — входные данные в момент времени t.
Массив y, где $y[t]$ — выходные данные в момент времени t.
f — обучаемая нейронная сеть.

Развернуть сеть f по времени на k соединенных друг за другом сетей;
repeat
  $x \leftarrow \vec{0}$;
  for $i \leftarrow 0$ **to** $n - 1$ **do**
    Передать на вход развернутой сети x, $a[t]$, $a[t+1]$, ..., $a[t+k-1]$;
    $p \leftarrow$ *распространить по развернутой сети входные данные*;
    $e \leftarrow y[t+k] - p$;
    Применить обратное распространение ошибки e *для развернутой сети*;
    Обновить все веса в развернутой сети;
    Усреднить веса по всем экземплярам сети f *в развернутой сети*;
    $x \leftarrow f(x)$;
  end
until *остановочное условие выполнено*;

Рис. 5: Алгоритм обратного распространения ошибки по времени

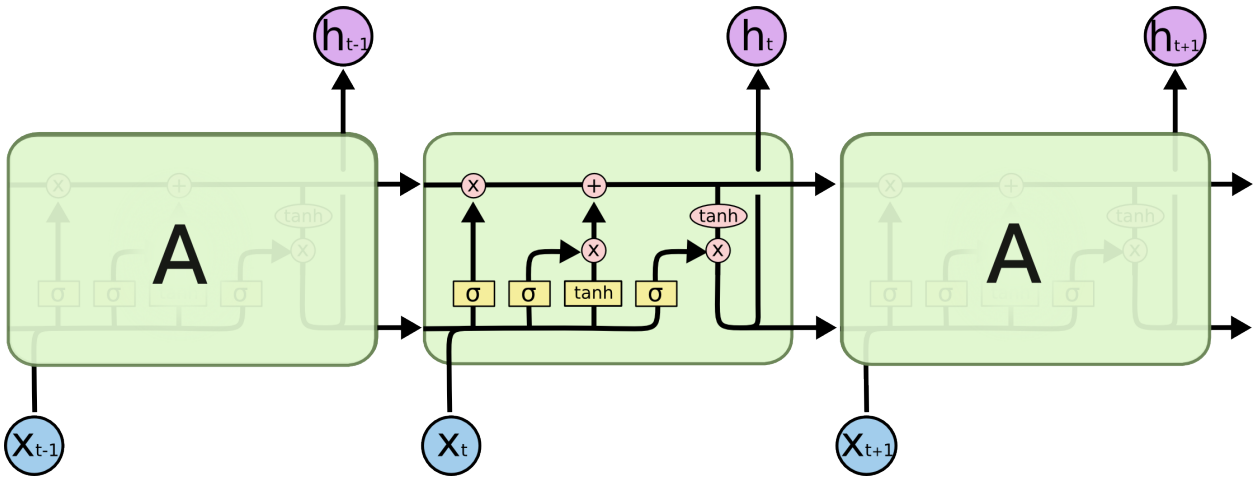

Рис. 6: Архитектура долгая краткосрочная память развернутая по времени

Рассмотрим эту архитектуру подробнее. Для начала введем определение сигмоидальной функции: $\sigma(x) = \frac{1}{1+e^{-x}}$. Далее, рассмотрим нейронную сеть с сигмоидальной функцией активации. Первый слой нейронной сети вычисляет множители к компонентнам вектора памяти.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

На втором шаге вычисляется новая информация, которая называется наблюдение и которая будет записана в память.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

$$\hat{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3}$$

На третьем шаге вычисляется линейная комбинация того, что находится в памяти, и наблюдения. Так получается новое состояние памяти.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{4}$$

На последнем шаге вычисляется значение выходного нейрона.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_0 C_t + b_o) \tag{5}$$

$$h_t = o_t * \tanh(C_t) \tag{6}$$

Полученные значения h_t и C_t передаются на вход нейронной сети в момент времни $t+1$. Для обучения нейронной сети используется рассмотренный ранее алгоритм обратного распространения ошибки по времени поскольку все приведенные выше функции являются дифференцируемыми.

2.3. Модель нейронной сети для задачи анализа тональности текста

Основная проблема обычной архитектуры LSTM заключается в том, что значения выражений 1, 2, 3, 4, 5 и 6 невозможно вычислить независимо друг от друга, что создает проблемы при попытках распараллеливания данной архитектуры. Для решения данной проблемы можно изменить архитектуру, заменив в ней формулу 5 на формулу 7.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$

Этого можно добиться не умаляя общности, если объединить все весовые матрицы входов в одну матрицу W, все весовые матрицы скрытых слоёв в одну матрицу U, а все векторы смещения в один вектор b. Тогда значения в нейронах находятся по следующей формуле:

$$z = W x_t + U h_{t-1} + b,$$

в этой формуле z состоит из i_t, f_t, $\hat{C}_t$ и o_t.

Данная архитектура является разновидностью LSTM и описывается формулами 1, 2, 3 и 7. Ее ключевая особенность заключается в том, что она поддается распараллеливанию, поскольку каждое из выражений не зависит от результатов других, в них есть зависимость лишь от результатов вычислений в предыдущий момент времени $t - 1$. Для дальнейшего удобства будем называть эту архитектуру LSTM-parallel.

Для избежания пробелмы переобучения используется техника dropout, подробнее описанная в работе [24]. Ее смысл заключается в том, что в зависмости от некоторой случайной величины(обычно — распределенной по закону Бернулли) на часть входов следующего слоя поступает 0 вместо значения, которое было передано с предыдущего слоя. В рассматриваемой моедли используется слой dropout со случайной величиной распределенной по закону Бернулли с вероятностью наступления события 0.5. Затем данные в нейронной сети попадают в слой усреднения(mean pooling), где усредняются между всеми, полученными на выходе предыдущего слоя, после чего передаются на вход в Softmax слой, который выполняет классификацию данных. Softmax слой — это такой слой, в котром используется следующая функция активации:

$$f(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{n} e^{z_i}}$$

На выходе данной модели будет величина $o \in [0, 1]$. Поскольку класифкация бинарная, то эту величину можно интерпретировать следующим образом: если $o < 0.5$, то классифицируемый текст относится к классу

«отрицательный отзыв», в противном случае — к классу «положительный отзыв».

Итоговая модель рекурретной неронной сети для задачи анализа тональности будет состоять из следующих слоев:

1. Входной слой.
2. Слой LSTM-Parallel.
3. Слой Dropout.
4. Слой усреднения.
5. Softmax слой.
6. Выходной слой.

Глава 3. Реализация

Описаные архитектура рекуррентной нейронной сети была реализованы на языке Python 2.7. Выбор языка обусловлен его популярностью в области машинного обучения и наличием большого числа библиотек. Реализацию можно разбить на 2 модуля: предварительная обработка данных и реализация нейронной сети.

3.1. Предварительная обработка данных

Этап предварительной обработки необходим, чтобы подготовить данные к классифкации или обучению. На этом этапе текст очищается от символов, которые не влияют на классификацию или не несут смысловой нагрузки в данной задаче. Например, ссылка в отзыве никак не влияет на эмоциональную окраску отзыва, но возможно влияет ее содержание. Для работы классификатора в случае английского языка необходимо преобразовать отрицательные чатицы, например заменить «aren't» на «are not» и т.п.

Предварительная обработка данных:

- удаление всех ссылок из текста;
- удаление всех символов emoji из текста;
- преобразование всего текста в нижний регистр;
- удаление неизвестных классификатору слов;
- удаление артиклей, предлогов и союзов из текста;
- преобразование отрицательных частиц;
- разбиение текста на слова с учетом знаков препинания.

3.2. Реализация нейронной сети

Для реализация была выбрана библиотека Theano [25]. Данная библиотека облегчает работу с символьными вычислениями и позволяет выполнять их как на CPU, так и на GPU, поддерживающих технологию Nvidia CUDA. Выполнение на GPU в задачах, в которых используются нейронные сети, позволяет получить производительность в несколько раз лучше, чем на CPU, как показано в работе [26].

Для решения задачи оптимизации в рамках обучения нейронной сети используется алгоритм AdaGrad [27], поскольку стохастический градиентный спуск обычно показывает плохие результаты в задачах, связанных с

анализом последовательностей, подробнее это описано в работе [28]. В качестве остановочного условия обучения был выбран метод ранней остановки, если результаты оставались неизменными или становились лишь хуже на протяжении 5 итераций обучения.

Реализация обучения:

- этап предварительной обработки данных;
- сохранение множества всех известных слов;
- преобразование обучающего набора в векторное представление мешок слов на основе множества известных слов;
- обучение рекуррентной нейронной сети методом обратного распространения ошибки по времени.

Классификация нового текста:

- этап предварительной обработки данных;
- преобразование текста в представление мешок слов на основе множества известных слов;
- распространение сигнала каждого слова друг за другом по нейронной сети;
- классификация текста по значению выходного нейрона. Если значение выходного нейрона меньше 0.5, то текст относится к классу «отрицательный отзыв», в противном случае — к классу «положительный отзыв».

Глава 4. Количественная оценка метода

Тестирование метода было проведено на Large Movie Review Dataset [17]. Для обучения нейронной сети использовались все 25000 примеров из набора данных, а для тестирования — 25000 примеров, предназначенных именно для тестирования. В таблице 2 представлены полученные результаты. До прекращения обучения нейронная сеть обучалась 31 итерацию.

Если сравнить результаты из таблицы 2, то можно заметить, что предложенная архитектура нейронной сети показала результат на 4% лучше существующих методов. Дополнительно в таблице 2 представлено время обучения рекуррентной нейронной сети на CPU и на GPU[8]. Использование GPU в данной задаче дает прирост производительности в ∼6 раз во время обучения, что довольно критично, если есть необходимость переобучения нейронной сети на новых примерах данных. Такая потребность может возникнуть, если осуществляется анализ тональности текстов из социальных сетей — нужно учитывать, например, сленг, который постоянно меняется.

Классификатор	F_1**-score**	**Время обучения**
Найвный байесовский классификатор	0.78	21 секунда
Логистическая регрессия	0.80	1382 секунды
Метод опорных векторов	0.80	2231 секунда
Рекуррентная нейронная сеть на CPU	0.84	8652 секунды
Рекуррентная нейронная сеть на GPU	0.84	1373 секунды

Таблица 2: Сравнение методов решения задачи анализа тональности текста с предложенной архитектурой нейронной сети

[8]Для тестирования на GPU использовалась видеокарта Nvidia GTX Titan X.

Выводы

В ходе данной работы были решены поставленные задачи и достигнуты следующие результаты:

1. Произведено сравнение существующих методов обучения с учителем: наивный байесовский классификатор, логистическая регрессия и метод опорных векторов. Подробнее о сравнение написано в разделе 1.3.
2. На основе нейронных сетей была предложена, обоснована и реализована архитектура рекуррентной нейронной сети для задачи анализа тональности текста. Подробнее об архитектуре нейронной сети написано в разделе 2.3, а о реализации — в главе 3.
3. Произведено сравнение нового метода с существующими. Сравнение производилось на одном и том же наборе данных Large Movie Review Dataset. Полученный метод показал улучшение в результатах классификации на 4%. Подробнее сравнение описано в главе 4.

Заключение

Задача анализа тональности текста является одной из наиболее интересных задач в обработке естественного языка на данный момент. До сих пор нет ни одного метода, который бы полностью решал данную задачу. Проблемы, которые ранее стояли перед исследователями, занимавшимися данной задачей, ушли в прошлое, на место им пришли новые. Скорее всего и в дальнейшем весь прогресс в этой задаче будет итеративным — исследователи будут решать одни проблемы, на смену которым придут другие. Так, например, одна из проблем в задаче анализа тональности текста сейчас — сарказм, поскольку все существующие методы не учитывает возможный саркастичный настрой автора отзыва. Выделение сарказма в тексте даже сформировалось в отдельную задачу классификации текстов. Это довольно сложная и интересная задача, поскольку даже люди не всегда могут распознать сарказм. Впрочем, в прошлом году эта задача была формализована и был описан подход к ее решению [29], который показывает хорошие результаты. За последние годы анализ текста на естественном языке продвинулся довольно далеко вперед, и не исключено, что в обозримом будущем подобные задачи будут полностью решены.

В качестве продолжения данной работы можно рассмотреть применение векторного представления слова word2vec [30] или sentence2vec [31] в качестве альтернативы векторного представления мешок слов. Эти представления позволили улучшить результат работы методов классификации в задаче классификации текстов, поэтому вполне возможно, что они позволят улучшить и результаты, полученные в данной работе. Также можно сформулировать и решить задачу классификации мнений на субъективные и объективные, дополнить полученную архитектуру неиронной сети так, чтобы одновременно решались и задача анализа тональности, и задача классификации мнений.

Список литературы

[1] Zhang Y., Jin R., Zhou Z.-H. Understanding bag-of-words model: a statistical framework. // International Journal of Machine Learning and Cybernetics, 2010. Vol. 1. P. 43 – 52.

[2] Boulanger-Lewandowski N., Bengio Y., Vincent P. Audio chord recognition with recurrent neural networks // Proceedings of the 14th International Society for Music Information Retrieval Conference, 2013. P. 335 – 340.

[3] LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition // Proceedings of the IEEE, 1998. Vol. 81, Issue 11. P. 2278 – 2324.

[4] Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks // Advances in neural information processing systems, 2012. P. 1097 – 1105.

[5] Watanabe Z., Erdogan S., Hershey H., Chen Z., Watanabe S., Erdogan H., John R. Hershey // Integration of Speech Enhancement and Recognition using Long-Short Term Memory Recurrent Neural Network, 2015.

[6] Chen X., Liu X., Gales M.J.F., Woodland P.C. Recurrent neural network language model training with noise contrastive estimation for speech recognition. // In Acoustics, Speech and Signal Processing (ICASSP), 2015. P. 5411 – 5415.

[7] Lai S., Xu L., Liu K., Zhao J. Recurrent Convolutional Neural Networks for Text Classification. // AAAI, 2015. P. 2267 – 2273.

[8] Turney P.D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews // Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002. P. 417 – 424.

[9] Pang B., Lee L., Vaithyanathan S. Thumbs up?: sentiment classification using machine learning techniques // Proceedings of the ACL-02 conference on Empirical methods in natural language processing. Association for Computational Linguistics, 2002. Vol. 10. P. 79 – 86.

[10] Tan S., Cheng X., Wang Y., Xu H. Adapting naive bayes to domain adaptation for sentiment analysis // Advances in Information Retrieval, 2009. P. 337 – 349.

[11] dos Santos C.N., Gatti M. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts // COLING, 2014. P. 69 – 78.

[12] Glorot X., Bordes A., Bengio Y. Deep sparse rectifier neural networks // International Conference on Artificial Intelligence and Statistics, 2011. P. 315 – 323.

[13] Murphy K.P. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). The MIT Press, 2012. ISBN: 0262018020.

[14] Manning C.D., Schutze H. Foundations of statistical natural language processing. MIT press, 1999.

[15] Nigam K., Lafferty J., McCallum A. Using maximum entropy for text classification // IJCAI99 workshop on machine learning for information filtering, 1999. P. 61 – 67.

[16] Tong S., Koller D. Support vector machine active learning with applications to text classification // The Journal of Machine Learning Research, 2002. Issue 2. P. 45 – 66.

[17] Maas A.L., Daly R.E., Pham P.T., Huang D., Ng A.Y., Potts C. Learning word vectors for sentiment analysis // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011. Vol. 1. P. 142 – 150.

[18] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research, 2011. Issue 12. P. 2825 – 2830.

[19] Elman J.L. Finding structure in time // Cognitive science, 1990. Vol. 14, no. 2. P. 179 – 211.

[20] Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. // International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6, Harvard, 1998. Issue 02. P. 107 – 116.

[21] Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[22] Werbos P.J. Backpropagation through time: what it does and how to do it. // Proceedings of the IEEE 78, Harvard, 1990. Issue 10. P. 1550 – 1560.

[23] Hochreiter S., Schmidhuber J. Long short-term memory // Neural computation 9, 1997. Issue 8. P. 1735 – 1780.

[24] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting // The

Journal of Machine Learning Research 15, Harvard, 2014. Issue 1. P. 1929 – 1958.

[25] Bergstra J., Breuleux O., Bastien F., Lamblin P., Pascanu R., Desjardins G., Turian J., Warde-Farley D., Bengio Y. Theano: a CPU and GPU math expression compiler // Proceedings of the Python for Scientific Computing Conference (SciPy), 2010.

[26] Oh K.S., Jung K. GPU implementation of neural networks // Pattern Recognition 37, 2004. Issue 6. P. 1311 – 1314.

[27] Zeiler M.D. ADADELTA: an adaptive learning rate method, 2012.

[28] Bengio Y., Simard P., Frasconi, P. Learning long-term dependencies with gradient descent is difficult. // IEEE Transactions on Neural Networks 5, 1994. Issue 2. P. 157 – 166.

[29] Rajadesingan A., Zafarani R., Liu H. Sarcasm detection on twitter: A behavioral modeling approach // Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, 2015. P. 97 – 106.

[30] Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space // Proceedings of Workshop at ICLR, 2013.

[31] Liu Y., Sun C., Lin L., Zhao Y., Wang X. Computing Semantic Text Similarity Using Rich Features, 2015.